Figure 6 is a flow chart of processing steps which a resource manager 121 can perform according to embodiments of the invention to allow a user 108 or a discover process 310 to create an object 302 within the object hierarchy 301.

In step 400, resource manager 121 receives an identification of a selected resource for which an object 302 is to be created in the resource hierarchy 301. As noted above, either the user 108 or to discover process 310 can provide such an identification of a resource.

Next, in step 401, the resource manager 121 queries the selected resource for any resource properties associated with that resource. Resource properties can include, for example, the type of resource (e.g., hardware, software, device, etc.), a serial number, name or other identification associated with the resource, or any other information which may be pertinent or relevant for the use of the resource management application 120.

In step 402, the resource manager 121 receives a simple name for the object 302 that will represent the resource. As an example, the resource manager 121 may use the serial number from the resource properties as the simple name 304 for the object 302. Alternatively, the user 108 may supply the simple name 304 as a text string. The simple name should be unique in its home context. If not, the resource manager 121 can prompt the user to enter a unique simple name for the object 302 that will represent the resource.

In step 403, the resource manager determines if the object 302 has a pre-selected or predetermined home location within the object hierarchy 301. This determination may be made, for example, using a specific resource property that indicates that the resource for which this object 302 represents is related to another particular resource. As an example, if the resource for which the object 302 represents is a disk drive in a data storage system, then perhaps the data storage system resource itself (i.e., an object 302 corresponding to the data storage system) serves as the home location in the object hierarchy 301 for the disk resource object 302. Generally, some resources may have predefined home locations that depend upon pre-existing relationships to other resources. For instance, software applications which are installed in a particular computer system may use an object 302 that represents that computer system in the object hierarchy 301 as their home 305. To illustrate the home concept with additional examples, disks and

volumes may have predefined homes of the storage systems in which the reside. Files may have a pre-defined home of the volume or partition in which they reside. Ports may have a home of a switch or router in which they reside, and so forth.

If the resource manager 121 determines in step 403 that no pre-selected or predetermined home 305 is assigned or is apparent for the resource for which the object 302 represents, processing proceeds to step 404 at which point the resource manager 121 receives a user selection of a home location for the object 302 within the object hierarchy 301. For example, the resource manager 121 can allow the user 108 to select a particular home object 302 within the object hierarchy 301 (e.g., via displaying the current hierarchy 301 in the graphical user interface 150) to serve as the home for the new object been created according to the processing steps in Figure 6.

After processing step 404, or, after the resource manager determines a pre-selected home location in step 403, processing proceeds to step 405 at which point the resource manager 121 determines if the home object located at the selected home location within the object hierarchy 301 is "transparent". As explained above, if an object 302 is indicated as being transparent 318 (Figure 4), then that object 302 cannot serve as a home object. As such, new child objects which are initially created or inserted below a transparent object in the object hierarchy 301 will not have their home 305 assigned to the transparent object. Rather, such new objects will be assigned a home which is equal to the first non-transparent object that is hierarchically above the transparent object within the object hierarchy 301. In this manner, transparent object can be used to categorize or group resources without requiring the group objects to be included in a home that defines a path to the object in the object hierarchy 301.

In step 405, if the resource manager 121 determines that the home object 302 at the selected home location in the object hierarchy 301 is transparent, processing proceeds to step 406 at which point the resource manager 121 upwardly traverses the object hierarchy 301 starting from the selected home object location (i.e., starting at the transparent object) until the resource manager 121 encounters the first non-transparent home object within the object hierarchy 301 thus identifying a proper home location to be that first non-transparent home object in the object hierarchy 301.

In step 405, if the home object at the selected home location is not transparent or, after processing step 406 to find the home to a non-transparent object that exists above the transparent object in the object hierarchy 301, the resource manager 121 proceeds to process step 407 at which point the resource manager 121 sets the home of the object 305 to be the current (i.e., identified) home location within the object hierarchy 301.

Next, in step 408 the resource manager 121 (i.e., under direction of the user 108) sets the object group 316, transparent 318, and terminal 320 properties for the newly created object 302. In other words, the resource manager 121 prompts the user 108 to indicate whether or not the new object 302 just created is to be a group object and if so, is the group object to be transparent or terminal. As noted above, if an object is indicated by a user 108 as being terminal, the actions performed by the user 108 on that object are not carried out on child objects that hierarchically relate below the terminal group object. This provides a mechanism for a user 108 to create a group object in the object hierarchy in order to categorize resources, for example. Then, if this group object is related for example to another group object which is acted upon in some manner, any child objects which depend from (i.e., that hierarchically relate below) the group object that is marked as terminal will be unaffected by those actions. Terminal group objects thus provide a security mechanism by which a user can place a representation of resources within the graphical user interface 150 without fear of accidentally manipulating, operating upon or otherwise changing any aspects of those resources.

Figure 7 is a flow chart of processing steps which the resource manager 121 performs according to one embodiment of the invention in order to traverse the object hierarchy 301 to produce representations of objects within a graphical user interface 150.

In step 470, the resource manager 121 begins an object hierarchy 301 tree traversal process. The resource manager 121 can use any type of tree traversal algorithm to obtain information concerning each object 302 within the object hierarchy 301.

Next, in step 471, the resource manager 121 enters an execution loop which operates upon each object 302 within the object hierarchy 301.